**Finding closest pair of points using closest pair algorithm**

#include<bits++.h>

using namespace std;

class point

{

public:

int x, y;

};

int compareX(const void\* a, const void\* b)

{

Point \*p1=(Point \*)a,\*p2 =(Point \*)b;

return (p1 ->x - p2 ->x);

}

int compareY(const void\* a, const void\* b)

{

Point \*p1=(Point \*)a,\*p2=(Point \*)b;

return (p1 ->y - p2 ->y);

}

float dist(Point p1,Point p2)

{

return sqrt( (p1.x - p2.x)\*(p1.x - p2.x)+(p1.y - p2.y)\*(p1.y - p2.y)

};

}

float bruteForce(Point P[ ],int n)

{

float min =FLT\_MAX;

for (int i=0;i<n;++i)

for(int j=i+1;j<n:++j)

if(dist(p[i],p[j])<min)

min=dist(P[i],p[j]);

return min;

}

float min(float x, float y)

{

return(x<y)? x:y;

}

float stripClosest(Point strip[ ],int size, float d)

{

float min=d;

qsort(strip,size,sizeof(point), compareY);

for(int i=0,i<size;++i)

for(int j=i+1;j<size && (strip[j].y-strip[i].y)<min;++j)

if(dist(strip[i],strip[j])<min)

min=dist(strip[i],strip[j]);

return min;

}

float closestUtil(Point P[ ],int n)

{

if(n<=3)

return bruteForce(P,n);

int mid=n/2;

Point midPoint=P[mid)];

float d1=closesUtil(P,mid);

float dr=closesUtil(P + mid,n - mid);

float d=min(d1,dr);

Point strip[n];

int j=0;

for(int i=0;i<n;i++)

if(abs(P[i].x-midPoint.x)<d)

strip[j]=p[i],j++;

return min(d,stripCloseset(P,n);

}

int main()

{

Point P[] {{2,3},{12,30},{40,50},{5,1}{12,10},{3,4}};

int n=sizeof(P)/sizeof(P[0]);

cout<<"The smallest distance is"<<closeset(P,n);

return 0;

}

Output:

